Misuse/Use Cases and Security Use Cases in Eliciting Security Requirements

Qian Gao
Department of Computer Science
The University of Auckland
qgao005@ec.auckland.ac.nz

Abstract

In recent years, Misuse Cases, concentrating on the threats and exceptions to a system, has become a popular method of analyzing and eliciting security (nonfunctional) requirements when combined with Use Cases. Additionally, some recent papers show that another kind of Use Cases, called Security Use Cases focusing on the harmful sequences, can help to elicit security requirements. In this paper we ask, and attempt to answer, the following questions. What is the relationship between Misuse/Use Cases and Security Use Cases? What are the differences between these two methods when applied into the same cases to specify security requirements? Are there any important distinctions between results? If they are applied together, can productivity be improved or not?

1. Introduction

Presently an increasing number of software development organizations recognize that developing security requirements is more important than designing protections because paying attention to security requirements in the early stages of the software lifecycle potentially saving millions of dollars. Thus, the methods analyzing and documenting security threats and security requirements, like Misuse / Use Cases and Security Use Cases, have become increasing necessary.

Misuse Cases, a new form of an old technology, are an extension of Use Cases. Use Cases work fairly well applied to the process of examining functional requirements. However, when used to elicit nonfunctional requirements, for example, security requirements, they do not work as successfully. Differing from Use Cases, Misuse Cases can be applied to specifying nonfunctional (security) requirements, as they are good at exploring exceptions and threats rather than functions within a system. The metaphor of Misuse/Use Cases described in [1][2] is adopted by this paper. This metaphor makes elicitation into a game, like Chess or Go, which works properly to show the eliciting process and the relationship between Misuse Cases and Use Cases.

In recent years, a closely related topic of research, called Security Use Cases, has been developed. Security Use Cases are another form of Use Cases specifically used to elicit security requirements. Compared with Misuse/Use Cases, Security Use Cases focus on the hostile actions from attacking agents as well; but they have different eliciting process and specified security requirements. To further reveal their differences, this paper is to compare these two methods by applying them into the same cases.

This paper begins by introducing Misuse/Use Cases, their metaphor, utilizations (eliciting security requirements) and a specific example. Section 3 talks about some opinions on the topic of Security Use Cases and explains this method through two examples in e-commerce [4]. In Section 4, according to the steps from [1][7], it is shown how Misuse/Use Cases can be applied into the cases introduced in Section 3. Section 5 compares these two methods and concludes the advantages and disadvantages of them each. Finally, Section 6 is the conclusion of this paper.

2. Misuse/Use Cases

2.1 Principle and Utilization

To understand Misuse Cases, it is necessary first to understand Use Cases. This is because Misuse Cases were developed from Use Cases and are based on similar

principles. Use Cases [8] are organized collections of scenarios based on the sequences of actions taken by normal users. That is to say, they focus on what a system can do or its purpose. Hence, Use Cases are suitable for analyzing and eliciting functional requirements in a system. This works through written specifications of a system's actions in textual form that includes the name of a use case, actor, precondition, flow of events, post-condition and an alternative path. Applying Use Cases increases the clarity and reusability of requirements.

With the development and growth of e-commerce and m-commerce people have continued to consciously and deliberately attack the systems that support these. This has lead to the growing importance in effective security requirements. Unfortunately, looking on the positive side of a system, Use Cases are deficient in eliciting these requirements. Thus, Use Cases have to be extended to Misuse Cases to adapt to this area. Misuse Cases [1][2][7], are a specialized form of Use Cases that focus on the exceptions and threats caused by hostile agents to a system. The simplest application of Misuse/Use Cases is to elicit specific security requirements in both informal and formal situations. Besides analyzing requirements, Misuse/Use Cases are also valuable in many other types of application [1][2]. Some of these include identifying exceptions, generating test cases and trading off conflicts when designing a system which are not explained in this paper.

Usually the approach of developing Misuse/Use Cases is top-down, which moves recursively from higher to lower levels; analyzing and specifying requirements from the points of both users and misusers. Existing Use Cases are continually challenged by potential threats, resulting in the discovered threat prompting new functions. The cases relationship explored at lower levels may be treated as the supplement of higher levels. The process will be repeated continuously until the problems in the system appear to be defined completely. The next part discusses the metaphor employed to explain a typical example.

2.2 Misuse/Use Cases Example [1]

Misuse/Use Cases employ a “game” metaphor whose algorithm is described as one team’s strategy of “thinking ahead to the other team’s best move and acting to block it” [1]. Using this metaphor derives several benefits. The metaphor is more comprehensive than expressing in words, it provides clear relationship between cases and finally it is easy to find and trade off conflicts. In this metaphor, users or hostile agents are represented by several stickmen; white and black teams respectively stand for Use Case and Misuse Case. The relationship is as follows: a system at a higher level ‘includes’ subsystems at lower levels; a misuse case 'threatens' a use case; a use case 'mitigates' a misuse case. The whole elicitation has a balanced zigzag form from up to down. As in the car example illustrated below, when a thief attempts to steal the car, the driver has to extend a subsystem called “lock” to defeat this attempt. The thief then tries to short the ignition which the diver can control by locking the transmission. However, do you believe that the thief will give up? Possibly not. Hence **the main constraint of this metaphor is that there is always the possibility of unknown exceptions when relying on Misuse/Use Cases.**

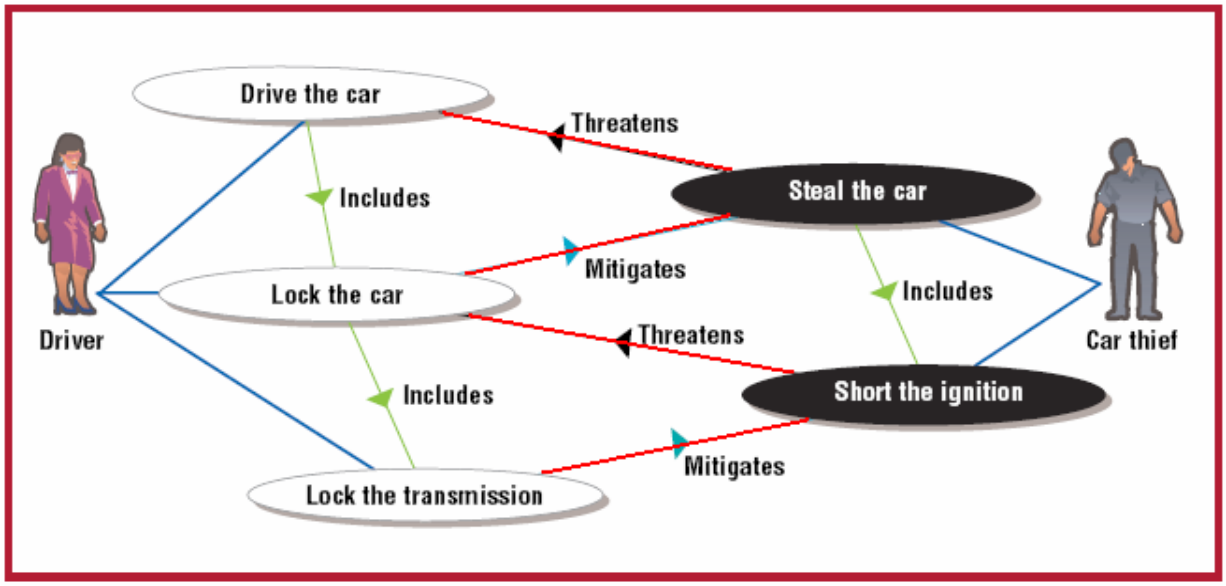

Fig.1. Misuse/Use Cases Elicitation for a Car System

3. Security Use Cases

3.1 Principle and Utilization

Security Use Cases are another kind of Use Cases particularly used to elicit security requirements whose generation and utilization are introduced in [3][4][5][6]. As explained Use Cases, the principle and metaphor of Security Use Cases can be omitted here. One approach to applying them is to separate the elicitation of security requirements (Security Use Cases) form that of functional requirements (Applications Use Cases). Security Use Cases require several abstracted security services with security conditions. When their security conditions are satisfied, they will extend the required functions (application Use Cases) to a system. Essentially, in different systems there are several general security requirements [3][4][6]. Thus, **essential Security Use Cases provide a highly reusable way of organizing and specifying security requirements. This is the main reason why in spite of the growing interest in Misuse Cases, this approach has been applied into large-scale industrial use.**

According to the threat analysis, six kinds of security services (Security Use Cases) are elicited in application systems. These are listed as follows:
"**Authentication** Service: allows an entity (a user or system) to identify itself positively to another entity.
Access Control: protects against unauthorized access to resources.
Confidentiality: protects against information being disclosed to any unauthorized entity.
Integrity: protects against unauthorized changes to data.
Non-repudiation: protects against one party to a transaction or communication activity later falsely denying that the transaction or activity occurred.
Availability (denial of service): Denial of service occurs when an authorized party is prevented from accessing a system to which he has legitimate access. [4]"

3.2 Security Use Cases Examples [4]

Before we start looking at them, there are some preconditions that will make the comparison more precise:

1. The examples are based on application system security rather than network security;
2. With the above precondition, denial of service does not need to be considered here;
3. The assumption is made that the database is secure.

3.2.1. Browse Catalog

The first use case is "browse catalog" where a customer can browse various catalogs and view various items from the catalogs. In the authors' opinion, an **access control** and **authentication** service should be applied into this case to improve security. Authentication identifies users through identity input, with which the system can accept legitimate users (identity information is kept for its access control) and refuse those invalid ones. If the customer is authenticated as a legitimate user, the system will use his identity information to check whether he has permission to access a specific catalog. As Security Use Cases agree to the metaphor used in Use Cases (textual form), they can be presented as:

"**Use Case Name: Browse Catalog**

Summary: Customer chooses a catalog to browse.

Actor: Customer

Precondition: System is ready.

Description (Base use case):

1. *<authentication>*
2. Customer chooses to view a catalog index.
3. System displays catalog index.
4. User requests to view a specific catalog.
5. *<access permission>*
6. System reads the catalog and displays it.

Postcondition: Customer has browsed a catalog. [4]"

Besides this, the use case “browse catalog” has another two extension use cases to define in the same form. These are “authentication” and “access permission”. However, their definitions do not need to be listed here because they are not necessary for our comparison. Figure 2 shows the relationship between these.

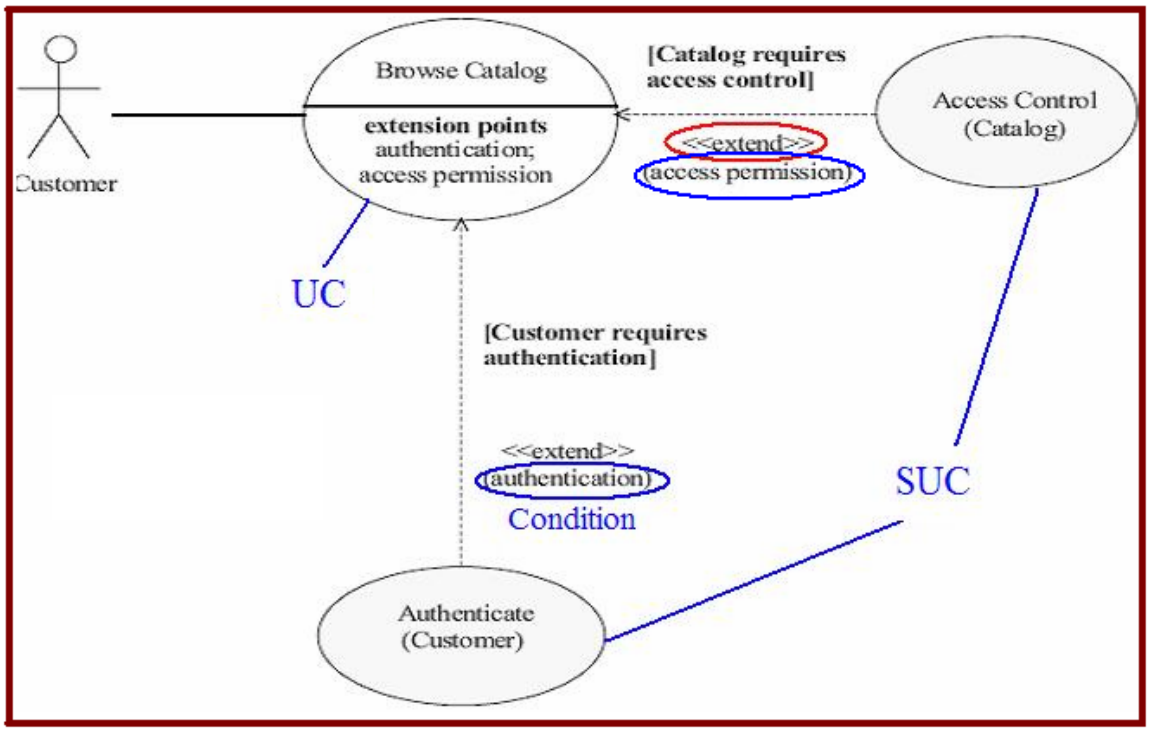

Fig.2. Browse Catalog with Security Use Case in Electronic Commerce System

3.2.2. Perform Purchase

The other example of Security Use Case application is used to perform secure purchase between customers and suppliers through purchase requests. During the purchase, a customer wants to send his purchase request to a supplier and pay by credit card. This causes the system to **encrypt** the request to maintain **confidentiality** and protect data from being changed with **integrity** check value. Besides this, to avoid falsely denying the request, the system needs a **non-repudiation** service. Usually digital signature is employed in this scenario. Figure 3 illustrates this process, which includes two application use cases (“Send Purchase Request” and “Receive Purchase Request”) and the above three security use cases. Unlike the first example, there are two operators of

one request, so that it is descried from two points. The descriptions of these use cases in textual form are omitted here, which define the order of applying the security use cases.

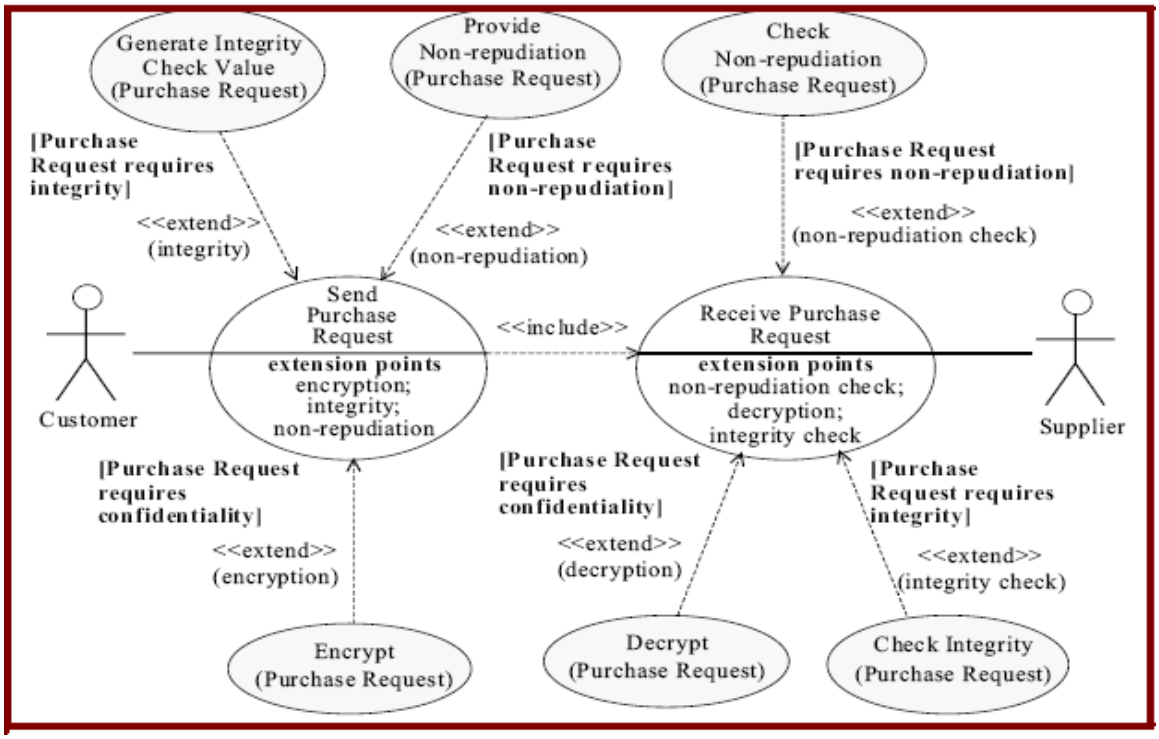

Fig.3. Perform Purchase Request with Security Use Case

4. Application of Misuse/Use Cases to e-commerce cases

4.1. Browse Catalog with Misuse/Use Cases

To apply Misuse/Use Cases to these two e-commerce cases, six students were invited. All having some security experience in the computer science department. A small informal workshop was formed in which we pretended to be not only users but negative agents threatening the system. Here Misuse/Use Cases were used as an informal approach to analyze and specify security requirements in small systems. Thus, the group just needed to analyze and list all of the misuse/use cases and their relationship according to the steps introduced in [1], but not to prepare for the later project. During this process, there were several iterations needed to refine the elicitation. In addition, the assumption was made that the misusers to be considered in these cases would only include outsiders instead of

insiders. This does not affect our result. For the “browse catalog” use case, our work is shown as:

1. **To explore what the system is and roughly think about its functions**: The system functions within some e-commerce catalogs whose main functions are providing catalog browsing and banning illegal reading.
2. **To clarify the users and misusers**: Users here should be **customers and the system**; misusers can be the users with intention for misuse or **hackers**. However, as there is no writing operation in the system at all, the users with the potential for misuse can be ignored here.
3. **To confirm and write down what the exact use cases are performed by users**: In our opinion, the use cases in this example are quite simple, basically identical to our estimation. In this situation, we can use one stickman to represent both of the two users to simplify the graph. In this scenario the elicited use cases are **browse catalog** and **security control**. Here, “Security control” is a sub-function of the main function which is a “browse catalog” function.
4. **To brainstorm a list of misuse cases for hostile agents**: Now we already have use cases, so we only needed to search for ways threatening them. Besides misuse cases, the relationship between these and the existing use cases should be established. According to our analysis, the main attempt of a hacker is to intrude into the catalog which is not open to him. The hacker can threaten the main use case through **brute-force password attack**, **system security-hole attack**, and **simulating users**. These three kinds of attacks concluded by our group members were thought to be more general and powerful than others.
5. **To consider how to mitigate the misuse cases**: There are several security mechanisms for each threat. Regardless, **Misuse Cases do not focus directly on the mechanisms**, **so any one of these can be selected to provide a general idea**. First we decided to use **log access attempts** to defeat a brute force attack; second to keep hackers from the system by **firewalls**; last to set **user recognition** if hackers pretend to be users. All of these use cases belong within the “security control” category.

6. **To document subsystem use cases and some misuse cases**: This dispensable process can be seen to supplement metaphor when a system is quite complex. Accordingly, we did not think it is necessary in this example.

The result of the process shown in Figure 4 gives us a more direct understanding of the above explanation.

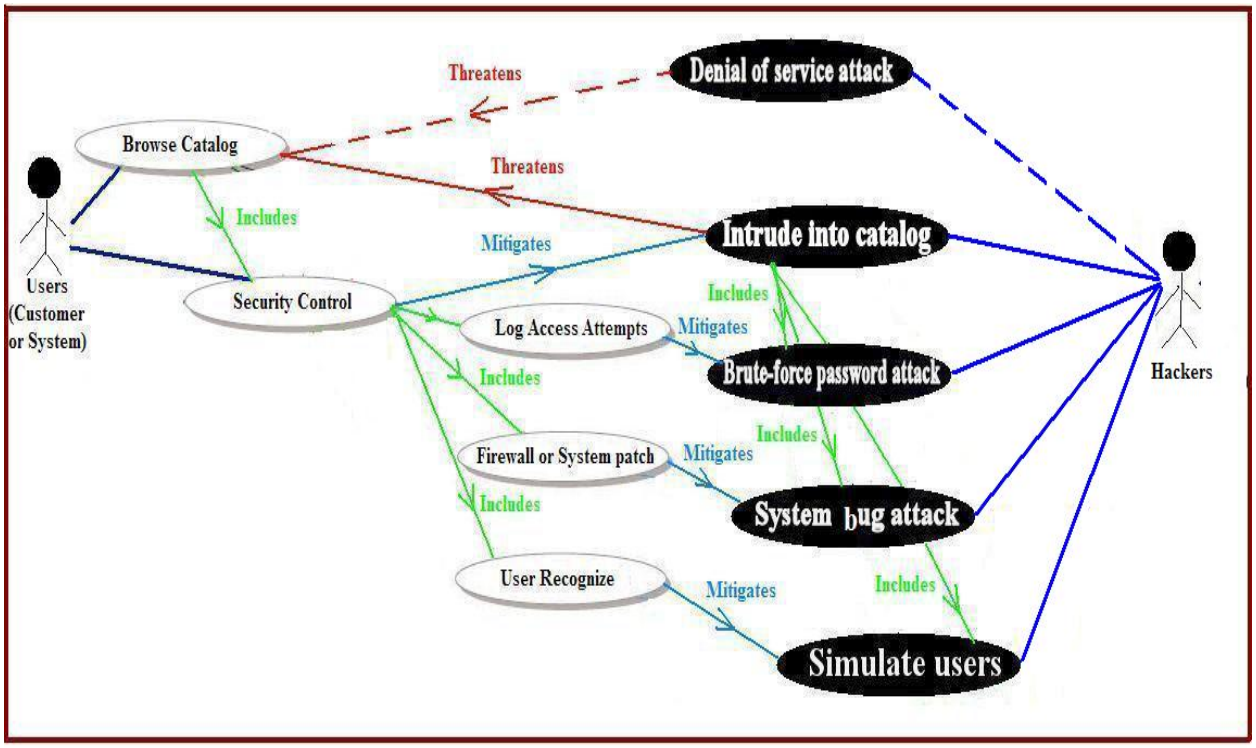

Fig.4. Browse Catalog with Misuse/Use Case

4.2. Perform Purchase with Misuse/Use Case

In order to make our comparison more precise and convincing, we applied Misuse/Use Cases into the other e-commerce case given in 3.2.2, "perform purchase". The main steps taken in this use case to ascertain security requirements are the same with those of the last application, which I summarize as follows:

The system suppliers try to provide customers with purchase service and security protection. The users are **customers and the system** including those who fill out purchase requests and pay by credit card. The misusers include the **users performing misuse cases** as they may falsely **deny their purchase requests**. As for **hackers** often attempt to **steal or change a user's information** evilly during purchase. Reflecting on

the threat, the security system applies several subsystems to block these threats. This involves a **required digital signature** for false denying, **encryption** for stealing, and **entry control** (e.g. hash function) for unauthorized change. However, hackers can use **decryption** to counter encryption, so that the system has to practice **more powerful encrypting mechanism**. The relationship between them is clearly displayed in Figure 5.

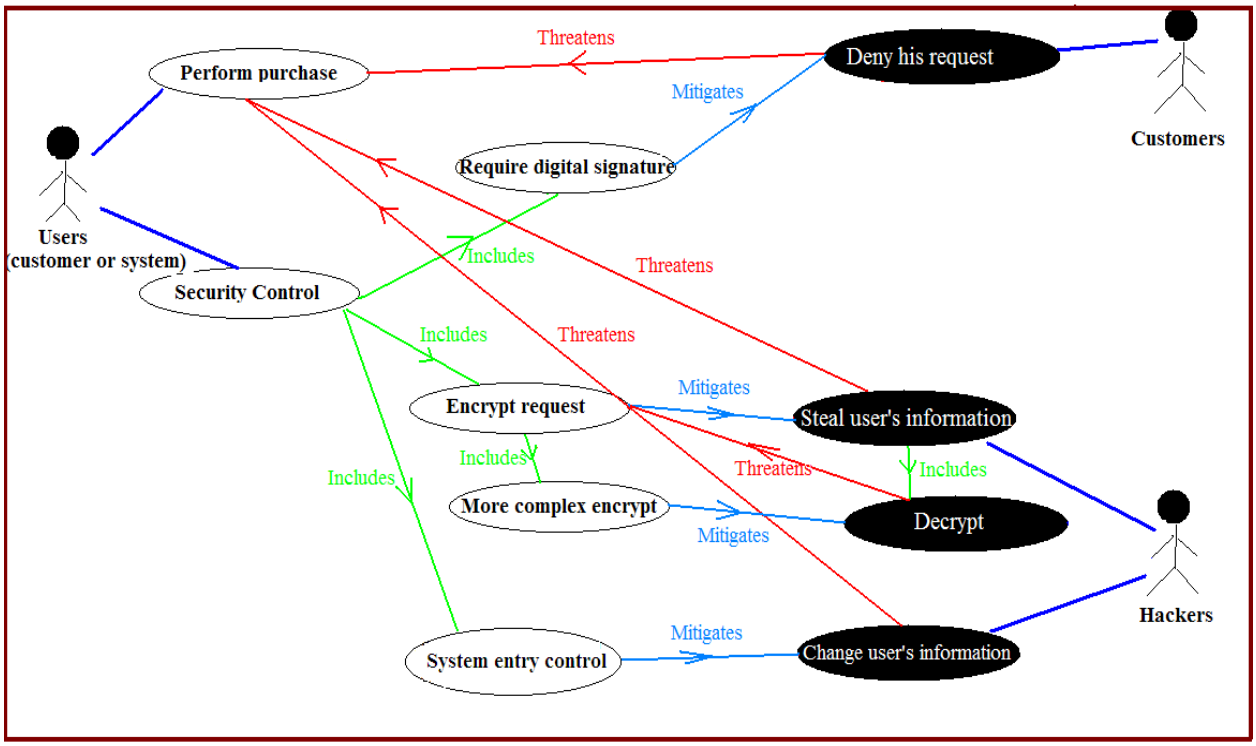

Fig.5. Perform Purchase Request with Misuse/Use Case

However, different from the first example, the specifications for subsystem use cases need to be written down as the performance of these use cases has a given order: "Encrypt request" should come after the other two requirements. This is because it also needs to encrypt the information of security. The following is the description for "Encrypt request":

Use Case Name: Encrypt request

Summary: System encrypts customer purchase requests.

Actor: System

Precondition: All of the information required by system (purchase and security) is ready.

Description:

1. System gathers the information from customers.
2. System encrypts the information.

Alternatives:
If the information of security is not ready, system has to wait until it is provided.
Postcondition: System has encrypted all the information.

5. Comparison

In this section, Figure 2 and 4, Figure 3 and 5 are compared. It details the similarities and distinctions between Misuse/Use Cases and Security Use cases from different aspects. The following is what was found:

(1). **Aim of Method**: Although both of the methods are useful in eliciting security requirements, they have totally different purposes. Because Misuse/Use Cases let misusers successfully attack applications, they are an effective way of analyzing security threats rather than directly specifying security requirements. In contrast, Security Use Cases allow applications to successfully protect themselves from relevant threats. These have the overt purpose of providing specific security requirements. Thus, **Security Use Cases (SUCs) are a more direct method than Misuse/Use Cases (MUCs) when eliciting security requirements**. From the point of view of the writer, **Security Use Cases are actually prompted by Misuse Cases**. There is however no display or specifications of their relationship when applying SUCs. In another words, the plots of SUCs stand for the left side (the "white team") in the MUCs' plots.

(2). **Metaphor Used**: For the first difference, MUCs must analyze threats from the perspectives of both users and hostile agents; while SUCs can only employ legal users. Moreover, the metaphors used by the two methods are quite different. On the one hand, both of them employ textual form for Use Cases; on the other hand, **MUCs treat the game diagram as the main metaphor, while SUCs use graphs as another expression of texts**. Hence the elicitation of MUCs is more enjoyable than that of SUCs.

(3). **Eliciting Result**: The requirements elicited from MUCs and SUCs are the most important thing for analysts to estimate and understand. From the four figures of the two

cases, we can see that the requirements for MUCs are more specific than those for SUCs. N**either of the two methods focuses on concrete security technologies**. Thus, **for most of the systems, SUCs are a highly reusable way of exploring security requirements. These are more effective and economical** as reuse of requirements could speed elicitation and lead to significant savings in development costs. However, it is hard to suggest completely reusable mechanisms for different systems. So **for some systems, MUCs still play an important role in eliciting security requirements**. However their potential to elicit the required result is restricted by the experience of analysts.

(4). **Conflicts Exploration**: MUCs provide conflict exploring by reacting when two cases conflict with each other in game diagrams. Through game diagrams the **conflicts revealed by MUCs become apparent**. Unlike MUCs, **SUCs use a more scientific and accurate way to define conflicts.** This is known as **Object Constraint Language (OCL), which can find the potential disagreements**. When a OCL is applied to "perform purchase" system, we may find that "Generate Integrity Check Value use case can have a mutually exclusive (zero or one) relationship with Provide Non-Repudiation if a digital signature for providing Non-repudiation security service is realized with a hash function, which provides an integrity service as well [4]". That is, if there are two mechanisms for one security service, conflicts will appear because they can have opposed decisions on the same request.

Overall, in the two examples, the requirements elicited through Misuse/Use Cases can be abstracted into those by using Security Use Cases. In most situations, Security Use Cases are more suitable for specifying security requirements. Nevertheless, Misuse/Use Cases can also work independently, and theoretically can be vital toward reinforcing the functions of Security Use Cases as they are able to analyze specific situations. There is a good example in [6] examining security requirements by combining Misuse and Security Use Cases.

6. Conclusion

This paper compares the performance of Misuse/Use Cases and Security Use Cases when applied to explore security requirements. Both of these two methods have their own advantages and disadvantages and are suitable for different kinds of applications. Obviously, Security Use Cases are more effective for the general large systems. For future work, it is valuable to think about employing the two methods together to improve their effectiveness.

7. References

[1] I. Alexander, "Misuse cases: use cases with hostile intent", *IEEE Software 20(1)*, 58-66, Jan/Feb 2003.

[2] I. Alexander, "Misuse cases help to elicit requirements non-functional", *Computing & Control Engineering Journal, Volume 14*, *Issue 1*, 40 – 45, Feb. 2003.

[3] Guttorm S., Donald F., Andreas L. Opdahl, "A Reuse-Based Approach to Determining Security Requirements", *Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality, REFSQ'03,* 127-136, 2003.

[4] Gomaa, H., Eonsuk Shin, M**.**, "Modelling complex systems by separating application and security concerns", *Engineering Complex Computer Systems, 2004. Proceedings. Ninth IEEE International Conference*, 19 – 28, 14-16 April 2004.

[5] Popp, G., Jurjens, J., Wimmel, G., Breu, R., "Security-critical system development with extended use cases", *Software Engineering Conference, 2003. Tenth Asia-Pacific 2003*, 478 – 487, 2003.

[6] Donald Firesmith, "Security Use Cases", *Journal of Object Technology, vol. 2, no. 3*, 53-64, May-June 2003.

[7] Sindre, G., Opdahl, A.L., "Eliciting security requirements by misuse cases", *Technology of Object-Oriented Languages and Systems, 2000, TOOLS-Pacific 2000, Proceedings. 37th International Conference on 20-23 Nov. 2000*, 120 – 131, 2000.

[8] I. Alexander, T. Zink, "Introduction to systems engineering with use cases", *Computing & Control Engineering Journal Volume 13, Issue 6*, 289 – 297, Dec. 2002.